Penerapan Algoritma *String Matching* untuk Mendeteksi Lagu yang Eksplisit

Jevant Jedidia Augustine - 13520133
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520133@std.stei.itb.ac.id

***Abstraksi*—Dengan adanya *music streaming platform* seperti Spotify, YouTube Music, Apple Music, dan lain-lain, setiap orang memiliki kebebasan yang besar untuk mendengarkan lagu yang mereka ingin dengarkan. Akan tetapi, seperti karya seni dan hiburan lainnya, beberapa lagu memiliki konten yang tidak pantas untuk didengar bagi anak-anak. Untuk mengatasi hal tersebut, *music streaming platform* mengimplementasikan fitur penyaringan sehingga lagu yang tidak pantas didengar bagi mereka yang di bawah usia (eksplisit) tidak akan muncul. Pendeteksian lagu yang tidak pantas dilakukan dengan memerika lirik dari lagu tersebut. Dengan menggunakan algoritma *string matching*, pendeteksian lirik lagu yang *explicit* dapat dilakukan dengan mudah.**

***Keywords—string matching;* eksplisit*;* lirik; lagu**

I. PENDAHULUAN

Musik atau lagu merupakan salah satu karya seni yang sudah lama ada. Musik itu sendiri adalah karya seni dimana adanya penataan suara dalam suatu waktu yang memiliki melodi, ritme, dan nada. Seiring dengan berjalannya atau bermainnya suatu musik, dapat dinyanyikan sebuah lirik dengan melodi, ritme, dan nada yang sesuai dengan musik yang mengiringi. Sebuah musik yang memiliki lirik dikenal sebagai lagu. Dengan adanya lirik, musik menjadi lebih mudah untuk diingat karena mereka yang mendengarkannya dapat bernyanyi bersama dengan penyanyi yang menyanyikan lagu tersebut. Adanya lirik pada suatu musik juga menyebabkan suatu musik untuk memiliki makna yang cukup konkrit dan mudah didapat oleh pendengarnya.

Dengan adanya lirik, ada banyak hal yang dapat disampaikan lewat lagu. Penulis lagu biasanya menggunakan lirik sebagai media untuk menampung dan menjabarkan hal-hal yang ingin disampaikan oleh penulis lagu itu sendiri. Penulis lagu dapat menceritakan sebuah cerita lewat lirik yang mereka tulis sehingga mereka yang mendengar lagu tersebut dapat membayangkan dirinya sedang mendengar sebuah cerita yang diceritakan oleh penyayi lagu. Penulis lagu juga dapat menyebarkan pesan seperti protes, kasih saying, ataupun perasaan yang dimiliki oleh penulis lagu lewat lirik lagu. Lagu menjadi salah satu media yang paling unggul dalam menyebarkan pesan karena pesan yang ingin dikemukakan oleh penulis lagu diiringi dengan musik yang dapat membuat pesan tersebut memiliki dampak yang lebih kuat.

Akibat dari lirik menjadi suatu elemen yang penting dalam lagu, beberapa musisi menggunakan bahasa yang cukup keras dalam lirik lagu mereka supaya pesan yang ingin mereka sampaikan dapat disampaikan dengan baik. Penggunaan bahasa yang keras tentunya menjadi masalah. Lagu yang menggunakan bahasa yang keras menyebabkan lagu tersebut menjadi kurang baik untuk didengar bagi mereka yang di bahwa umur. Lagu yang menggunakan bahasa kasar juga biasanya dianggap kurang sopan.

Eksplisitas dari suatu lagu menjadi perhatian bagi pihak penyiaran lagu seperti radio. Pada kebanyakan kasus, lagu-lagu yang dimainkan atau disiarkan adalah lagu yang tidak eksplisit karena lagu yang dimainkan dan disiarkan akan didengar oleh orang banyak, termasuk anak-anak. Seiring berjalannya waktu, *music streaming platform* seperti Spotify, YouTube Music, dan Apple Music menjadi sarana yang paling banyak digunakan oleh orang banyak untuk mendengarkan musik.

Gambar 1.1 Logo Spotify

Sumber: https://en.wikipedia.org/wiki/Spotify

Dengan adanya sarana-sarana tersebut, setiap orang dapat mendengarkan lagu yang mereka ingin dengarkan dengan bebas. Hal tersebut juga berarti pengaksesan lagu dengan konten yang eksplisit menjadi sangat mudah. Anak-anak dapat mengakses lagu dengan konten yang eksplisit tanpa sepengetahuan pihak yang bertanggung jawab. Untuk mengatasi hal tersebut, *music streaming platform* memiliki fitur untuk menyembunyikan lagu-lagu yang eksplisit. Untuk mengetahui lagu apa saja yang eksplisit, dilakukan pemeriksaan terhadap lirik dari lagu yang ada pada *platform* mereka dan lagu yang dianggap memiliki konten eksplisit akan ditandai.

Pada makalah ini, penulis akan membahas penggunaan algoritma *string matching* untuk memeriksa apakah lirik dari sebuah lagu mengandung konten yang eksplisit atau tidak.

II. TEORI DASAR

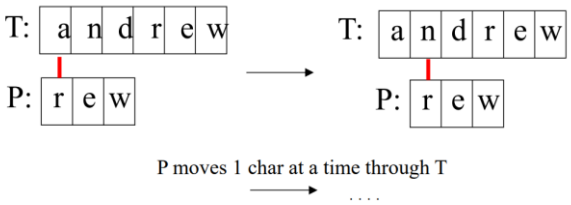
A. *String Matching*

String Matching adalah algoritma yang digunakan untuk mencari suatu pola yang terdapat pada sebuah teks. Teks pada kasus ini adalah *string* yang panjangnya n karakter, sedangkan pola atau *pattern* yang dimaksud adalah *string* dengan panjang *m* karakter yang akan dicari di teks dengan asumsi panjang karakter pola lebih kecil bila dibandingkan dengan panjang karakter teks (m <<< n).

Ada beberapa jenis algoritma yang dapat digunakan untuk menerapkan *string matching*, antara lain:

1. Algoritma *Brute Force*

Algoritma *brute force* akan memeriksa setiap posisi pada teks untuk mencari tahu apakah terdapat pola yang dimulai pada posisi tersebut.



Gambar 2.1 Ilustrasi *string matching* dengan algoritma *brute force*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Pencarian pola P dengan panjang m pada teks T dengan panjang n dimulai dari T[0]. Apabila karakter pada T[0] tidak sama dengan karakter pada P[0], maka pola akan digeser ke kanan sebanyak 1 karakter kemudian proses pencocokan akan diulangi. Apabila karakter T[0] sama dengan karakter P[0], maka dilakukan pemeriksaan terhadap setiap karakter pada T[0..m] dengan P[0..m]. Apabila setiap karakter sama, maka pola pada teks telah ditemukan, bila tidak, maka proses pencocokan akan diulangi lagi dengan pola yang digeser ke kanan sebanyak 1 karakter. Proses *string matching* akan selesai apabila indeks dari teks sama dengan n – m + 1 atau pola sudah ditemukan.

```
Teks: NOBODY NOTICED HIM
Pattern: NOT

  NOBODY NOTICED HIM
1 NOT
2  NOT
3   NOT
4    NOT
5     NOT
6      NOT
7       NOT
8        NOT
```

Gambar 2.2 Contoh dari *string matching* dengan algoritma *brute force*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

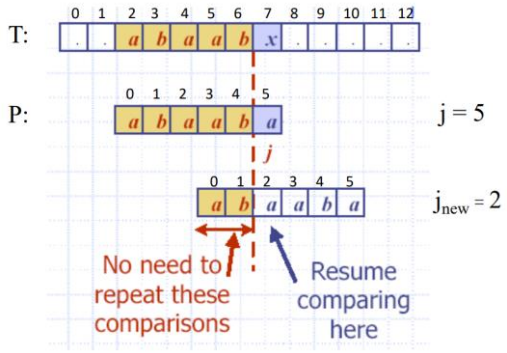
Kompleksitas dari *string matching* dengan algoritma *brute force* adalah:

- *Worst case scenario*: O(mn)
- *Best case scenario*: O(n)
- *Average case scenario*: O(m+n)

Algoritma *brute force* cepat apabila ruang lingkup karakter dari pola dan teks besar dan lambat apabila ruang lingkup karakter dari pola dan teks kecil.

2. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP merupakan algoritma *string matching* yang mirip dengan algoritma *brute force*, perbedaannya adalah pergeseran yang dilakukan oleh algoritma KMP lebih cermat daripada algoritma *brute force*. Apabila terdapat ketidakcocokan antara P[j] dan T[i], maka besar pergeseran pola yang dilakukan adalah besar prefiks terbesar dari P[0..j-1] yang juga merupakan sufiks dari P[1..j-1]. Hal tersebut dilakukan untuk mengurangi perbandingan karakter yang sia-sia.



Gambar 2.3 Ilustrasi *string matching* dengan algoritma KMP

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

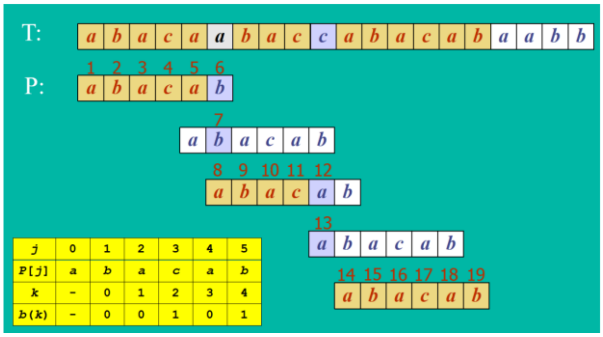
Jumlah pergeseran yang dilakukan oleh pola apabila terdapat ketidakcocokan dihitung dengan *border function (failure function)*. *Border function* b(k) didefinisikan sebagai ukuran terbesar prefiks dari P[0..k] yang juga merupakan sufiks dari P[i..k], dimana k merupakan posisi karakter pada pola sebelum ketidakcocokan (ketidakcocokan terjadi di P[j], k = j-1).

j	0	1	2	3	4	5	6	7	8	9
P [j]	a	b	a	b	a	b	a	b	c	a
k	-	0	1	2	3	4	5	6	7	8
b[k]	-	0	0	1	2	3	4	5	6	0

Gambar 2.4 Contoh dari *border function*

Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Algoritma KMP akan menggunakan *border function* untuk meningkatkan efisiensi pergeseran yang terjadi di algoritma *brute force*. Apabila terjadi ketidakcocokan pada P[j] dan T[i], maka P[j] akan diubah menjadi P[b(k)] dimana k adalah j-1, setelah itu pencarian dan pencocokan pola akan dilanjutkan.



Gambar 2.5 Contoh dari *string matching* dengan algoritma KMP

Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Kompleksitas dari algoritma KMP adalah:

- Menghitung *border function*: O(m)
- Pencarian string: O(n)
- Kompleksitas waktu total: O(m+n)

Kelebihan dari algoritma KMP adalah algoritma tidak perlu untuk bergerak mundur pada input teks. Akan tetapi, algoritma KMP kurang baik apabila ruang lingkup karakter pola dan teks besar.

3. Algoritma Boyer-Moore (BM)

String matching dengan algoritma Boyer-Moore didasari oleh 2 teknik.

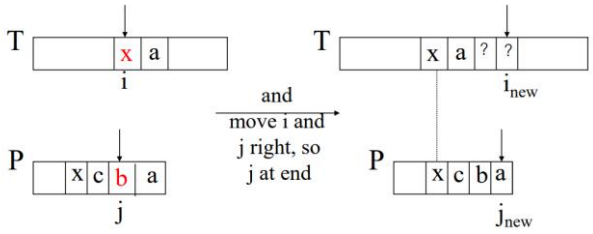
- Teknik *looking-glass*

  Mencari pola P pada teks T dengan bergerak mundur melalui P yang dimulai dari ujung P.

- Teknik *character-jump*

  Saat terjadi ketidakcocokan pada P[j] dan T[i] dan T[i] == x, ada 3 kasus yang mungkin terjadi.

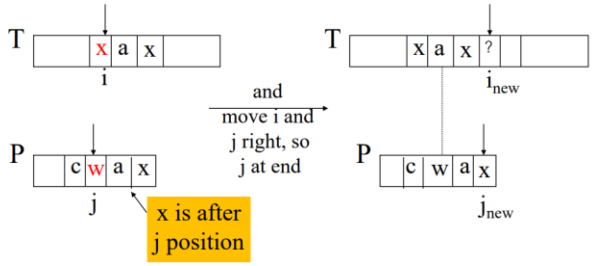
  - Kasus 1: P mengandung karakter x, maka geser P ke kanan sehingga x yang terakhir muncul sejajar dengan T[i].



Gambar 2.6 Teknik *character-jump* untuk kasus 1

Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

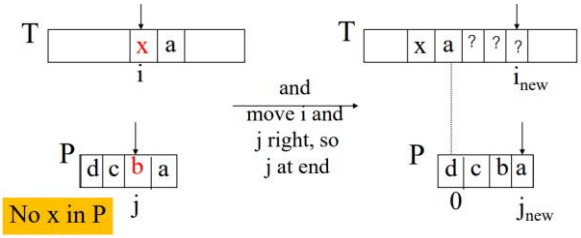
  - Kasus 2: P mengandung karakter x tetapi tidak dapat dilakukan pergeseran ke kanan untuk meluruskan x dengan T[i], maka geser P ke kanan sebanyak 1 karakter.



Gambar 2.7 Teknik *character-jump* untuk kasus 2

Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

  - Kasus 3: Bila kasus 1 dan 2 tidak berlaku, maka geser P sehingga P[0] sejajar dengan T[i+1].



Gambar 2.8 Teknik *character-jump* untuk kasus 3

Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Untuk mempermudah pergeseran dari *character-jump*, maka algoritma BM akan menggunakan fungsi *last occurrence*. Fungsi *last occurrence* L(x) didefinisikan sebagai indeks terbesar dari P dimana P[i] == x. Apabila indeks tidak ditemukan, maka nilai L(x) adalah -1.

Contoh dari penggunaan fungsi *last occurrence* pada pola "abacab" dan ruang lingkup karakter {a, b, c, d}:

x	***a***	***b***	***c***	***d***
L(x)	4	5	3	-1

Gambar 2.9 Contoh penggunaan fungsi *last occurrence*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Algoritma Boyer-Moore merupakan algoritma yang cepat apabila ruang lingkup karakter dari pola dan teks besar akan tetapi lambat apabila ruang lingkup karakternya kecil.

4. Regular Expression (Regex)

Regular Expression atau yang biasa dikenal sebagai regex merupakan salah satu metode *string matching* yang paling unggul. Regex itu sendiri dapat merupakan sebuah pola yang eksak maupun notasi yang membentuk suatu pola yang akan dicari. Penulisan notasi regex menggunakan simbol-simbol yang telah ditetapkan.

Character classes	
.	any character except newline
\w \d \s	word, digit, whitespace
\W \D \S	not word, digit, whitespace
[abc]	any of a, b, or c
[^abc]	not a, b, or c
[a-g]	character between a & g
Anchors	
^abc$	start / end of the string
\b	word boundary
Escaped characters	
\. * \\	escaped special characters
\t \n \r	tab, linefeed, carriage return
\u00A9	unicode escaped ©
Groups & Lookaround	
(abc)	capture group
\1	backreference to group #1
(?:abc)	non-capturing group
(?=abc)	positive lookahead
(?!abc)	negative lookahead
Quantifiers & Alternation	
a* a+ a?	0 or more, 1 or more, 0 or 1
a{5} a{2,}	exactly five, two or more
a{1,3}	between one & three
a+? a{2,}?	match as few as possible
ab|cd	match ab or cd

Gambar 2.10 Notasi-notasi dasar regex

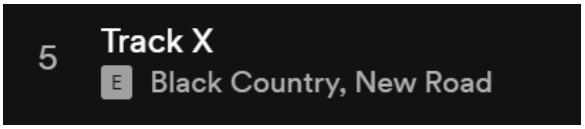
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Modul-Praktikum-NLP-Regex.pdf

B. Konten Eksplisit dalam Lagu

Bila merujuk kepada berbagai *music streaming platform*, konten eksplisit dalam sebuah lagu didefinisikan sebagai lirik lagu yang mengandung kata kasar (1 kata saja sudah cukup), referensi kepada kekerasan, pelecehan fisik maupun mental, referensi ke perlakuan seksual, serta penggunaan bahasa yang diskriminatif. Secara umum, sebuah lagu dikatakan memiliki konten yang eksplisit apabila lirik dari lagu tersebut tidak cocok untuk didengar oleh anak-anak. Perlu diperhatikan bahwa eksplisitas sebuah lagu dilihat dari liriknya, sehingga elemen-elemen lain dari lagu seperti komposisi dan instrumen tidak akan menentukan apakah sebuah lagu dikategorikan sebagai konten eksplisit atau tidak.

Gambar 2.11 Contoh lagu pada Spotify yang tidak eksplisit

Sumber: Spotify



Gambar 2.12 Contoh lagu pada Spotify yang ditandai eksplisit

Sumber: Spotify

Beberapa *music streaming platform* memiliki toleransi yang lebih tinggi daripada yang lainnya, sehingga satu lagu yang sama dapat dianggap eksplisit oleh suatu *music streaming platform* dan dianggap tidak eksplisit oleh yang lainnya.

Music streaming platform tetap membiarkan musik-musik eksplisit berada pada platform mereka karena mereka ingin lagu yang dirilis pada platform tersebut sesuai dengan keinginan musisi yang merilis lagu tersebut. Walaupun demikian, untuk menghindari konten eksplisit didengar dan dikonsumsi oleh anak-anak, *music streaming platform* memiliki opsi untuk menyembunyikan lagu-lagu dengan konten eksplisit.

III. Proses Pendeteksian Konten Eksplisit dalam Lagu

Untuk mendeteksi konten eksplisit yang berada di dalam lirik lagu, akan ditetapkan terlebih dahulu apa yang menyebabkan sebuah lirik lagu memiliki konten eksplisit. Akan digunakan definisi yang diberikan oleh berbagai *music streaming platform* sebagai acuan. Sebuah lagu akan dikatakan memiliki konten yang eksplisit apabila lagu tersebut memiliki lirik yang tidak cocok untuk didengar oleh anak-anak.

Berdasarkan definisi tersebut, maka dapat disimpulkan bahwa sebuah lagu akan ditandai eksplisit apabila lirik lagu tersebut mengandung kata-kata kasar. Hal tersebut juga dapat dibuktikan dengan melihat lagu-lagu yang ditandai eksplisit oleh berbagai *music streaming platform*. Lagu-lagu yang ditandai eksplisit mengandung kata-kata yang kasar di dalam liriknya. Walaupun berbagai *music streaming platform* menyatakan bahwa apabila lirik lagu mengandung referensi kepada kekerasan, pelecehan fisik maupun mental serta perlakuan seksual, kebanyakan lirik yang mereferensikan topik-topik tersebut akan mengandung kata-kata yang kasar sehingga untuk kasus kali ini, bisa disimpulkan bahwa apabila suatu lirik lagu mengandung kata-kata yang tidak pantas untuk didengar oleh anak-anak, maka lagu tersebut akan ditandai eksplisit.

Untuk mendeteksi kata kasar yang berada di dalam lirik lagu, perlu diketahui terlebih dahulu kata apa saja yang dikategorikan sebagai kata kasar. Kata-kata tersebut kemudian dapat disimpan dalam bentuk array, apabila jumlahnya hanya sedikit, atau disimpan di dalam *database*, apabila jumlahnya

banyak. Keempat algoritma *string matching* yang telah dikemukakan pada bagian dasar teori bisa digunakan untuk pendeteksian konten eksplisit pada lagu. Untuk penulisan makalah ini, penulis memilih untuk menggunakan algoritma BM untuk mendeteksi lagu yang eksplisit.

Algoritma dasar pendeteksi lagu eksplisit adalah sebagai berikut:

```
dict : array of String //array kata-kata kasar
lyric : String //lirik lagu
eksplisit = deteksiEksplisit(dict, lyric)
If eksplisit then
   Lagu ditandai eksplisit
Else
   Lagu tidak ditandai eksplisit
```

```
function deteksiEksplisit (dict : array of String, lyric : String) -> Boolean
for word in dict:
   if stringMatching(word, lyric) then
      return True
return False
```

Program utama akan menggunakan fungsi deteksiEksplisit untuk menentukan apakah sebuah lagu merupakan lagu eksplisit atau tidak. Fungsi deteksiEksplisit akan menerima masukan berupa *array of string* yang merupakan daftar kata-kata kasar dan *string* yang merupakan lirik dari sebuah lagu. Pada fungsi deteksiEksplisit akan dilakukan iterasi terhadap setiap kata yang berada di *array of string*. Pada setiap iterasi, akan digunakan fungsi stringMatching (dengan algoritma yang dipilih) dengan masukan kata (pola) dan lirik lagu (teks). Apabila kata ditemukan di dalam lirik lagu, maka iterasi akan selesai dan fungsi akan mengembalikan True. Bila setiap kata di dalam daftar kata tidak ditemukan pada lirik lagu, fungsi akan mengembalikan False. Program utama akan menandakan lagu eksplisit apabila hasil dari deteksiEksplisit merupakan True dan tidak eksplisit apabila hasil dari deteksiEksplisit adalah False.

IV. Implementasi

Implementasi dari algoritma pendeteksi lagu eksplisit dibuat oleh penulis menggunakan bahasa Python.

A. *Input* Lirik

Untuk mendapatkan masukan lirik dan mengubahnya menjadi *string*, akan digunakan fungsi readFile. Fungsi readFile akan mendapatkan masukan berupa nama file (.txt) yang berisikan lirik dari lagu yang akan diubah menjadi teks. Apabila isi dari file masukan berupa *multiline*, maka fungsi akan konkatenasi setiap *line* dan menggabungkannya menjadi 1 *string*.

```python
def readFile(filename):
    lyrics = ""
    with open(filename, encoding="utf8") as file:
        for line in file:
            lyrics = lyrics + " " + line.rstrip()
    return lyrics
```

Gambar 4.1 Fungsi readFile

Sumber: Penulis

B. Fungsi *buildLast*

Fungsi *buildLast* akan menerima masukan berupa *string* yang merupakan pola dan akan membangun sebuah *array* yang berisikan indeks kemunculan terakhir semua karakter yang ada di pola pada pola yang dimasukkan. Akan diinisialisasi sebuah *array of integer* (last) yang berukuran 128 dengan isi -1. Indeks kemunculan terakhir karakter *x* kemudian akan dimasukkan ke *array* last. Penempatan dari indeks tersebut bergantung dengan nilai ASCII dari karakter *x*. Apabila nilai ASCII dari karakter berada diantara 65 dan 90, maka indeks kemunculan terakhir akan ditempatkan di last[ASCII] dan last[ASCII+32], sedangkan apabila nilai ASCII karakter berada diantara 97 dan 122, maka indeks kemunculan terakhir akan ditempatkan di last[ASCII] dan last[ASCII-32]. Hal tersebut dilakukan karena pembangunan *array* dari *last occurrence* ini *case-insensitive*.

```python
def buildLast(pattern):
    last = []
    for i in range (128):
        last.append(-1)
    for i in range (len(pattern)):
        if 65 <= ord(pattern[i]) <= 90:
            last[ord(pattern[i])] = i
            last[ord(pattern[i])+32] = i
        elif 97 <= ord(pattern[i]) <= 122:
            last[ord(pattern[i])] = i
            last[ord(pattern[i])-32] = i
    return last
```

Gambar 4.2 Fungsi buildLast

Sumber: Penulis

C. Fungsi BM

Fungsi BM akan menerima masukan sebuah teks dan sebuah pola dan memeriksa apakah pola yang dimasukkan berada di dalam teks yang dimasukkan. Algoritma dari fungsi BM sama dengan yang telah dituliskan di bagian teori dasar. Pemeriksaan karakter pada fungsi BM bersifat *case-insensitive*. Fungsi BM akan mengembalikan True bila pola terdapat di dalam teks dan False bila pola tidak terdapat di dalam teks.

```
def BM (text, pattern):
    last = buildLast(pattern)
    n = len(text)
    m = len(pattern)
    i = m-1
    j = m-1

    if i > n-1:
        return False

    while i <= n-1:
        if pattern[j].lower() == text[i].lower():
            if j == 0:
                return True
            else:
                i -= 1
                j -= 1
        else :
            lo = last[ord(text[i])]
            i = i + m - min(j, 1+lo)
            j = m - 1
    return False
```

Gambar 4.3 Fungsi BM

Sumber: Penulis

D. Fungsi *checkExplicit*

Fungsi *checkExplicit* akan menerima masukan berupa nama file yang berisikan lirik yang akan diperiksa dan *array of string* yang berisikan kata-kata yang eksplisit. Akan dilakukan iterasi terhadap setiap kata yang berada pada *array of string* masukan kemudian akan dilakukan *string matching* terhadap kata dan lirik. Fungsi akan mengembalikan True bila satu atau lebih kata yang eksplisit berada di lirik dan False bila tidak dideteksi kata-kata eksplisit di dalam lirik.

```
def checkExplicit(filename, dict):
    text = readFile(filename)
    for word in dict:
        if BM(text, word):
            return True
    return False
```

Gambar 4.4 Fungsi checkExplicit

Sumber: Penulis

E. Program Utama

Program utama akan menampung daftar kata-kata kasar dalam bentuk *array* dan menerima masukan user yang berupa nama file yang berisikan lirik yang akan diperiksa. Program kemudian akan memanggil fungsi checkExplicit. Apabila hasil fungsi tersebut adalah True, maka lirik lagu yang dimasukkan eksplisit, bila False, maka lirik lagu tidak eksplisit.

```
dict = ["bitch", "shit", "fuck"]
filename = input("Masukkan nama file: ")
if checkExplicit(filename, dict):
    print("Lagu mengandung konten eksplisit")
else:
    print("Lagu tidak mengandung konten eksplisit")
```

Gambar 4.5 Program utama

Sumber: Penulis

V. Uji Coba

Uji coba akan dilakukan terhadap 4 lagu, 24K Magic – Bruno Mars, Soil – System of a Down, only shallow – my bloody valentine, dan The Dripping Tap – King Gizzard & The Lizard Wizard. Semua lirik yang akan digunakan untuk uji coba diambil dari genius.com.

- 24k Magic

Bisa dilihat pada lirik yang tercantum di genius.com bahwa lagu "24k Magic" mengandung konten yang eksplisit. Sehingga program diharapkan dapat mendeteksi bahwa lagu "24k Magic" merupakan lagu yang eksplisit.

[Verse 2]
Second verse for the hustlas (hustlas)
Gangstas (gangstas)
Bad bitches and ya ugly ass friends (Haha)
Can I preach? (Uh oh) Can I preach? (Uh oh)

Gambar 5.1 Lirik eksplisit "24k Magic"

Sumber: https://genius.com/Bruno-mars-24k-magic-lyrics

```
PS C:\ITB\Informatika\Sem 4\Stima\Makalah> python -u "c:\ITB\Informatika\Sem 4\Stima\Makalah\test.py"
Masukkan nama file: 24KMagic.txt
Lagu mengandung konten eksplisit
```

Gambar 5.2 Uji coba 1

Sumber: Penulis

- Soil

Lirik pada lagu "Soil" mengandung konten yang eksplisit, sehingga program diharapkan dapat mendeteksi bahwa lagu tersebut merupakan lagu yang eksplisit.

[Chorus: Serj Tankian]
Friends for years images in red
Blew off his own motherfucking head
Confidence, death, insecurity
The men fall unrealized

Gambar 5.3 Lirik eksplisit "Soil"

Sumber: https://genius.com/System-of-a-down-soil-lyrics

```
PS C:\ITB\Informatika\Sem 4\Stima\Makalah> python -u "c:\ITB\Informatika\Sem 4\Stima\Makalah\test.py"
Masukkan nama file: Soil.txt
Lagu mengandung konten eksplisit
```

Gambar 5.4 Uji coba 2

Sumber: Penulis

- only shallow

Lirik pada lagu "only shallow" tidak mengandung konten yang eksplisit sehingga program diharapkan tidak mendeteksi adanya konten yang eksplisit pada lagu "only shallow".

```
PS C:\ITB\Informatika\Sem 4\Stima\Makalah> python -u "c:\ITB\Informatika\Sem 4\Stima\Makalah\test.py"
Masukkan nama file: OnlyShallow.txt
Lagu tidak mengandung konten eksplisit
```

Gambar 5.5 Uji coba 3

Sumber: Penulis

- The Dripping Tap

Tidak ada konten eksplisit pada lirik "The Dripping Tap" apabila dilihat pada genius.com sehingga program diharapkan

dapat mengatakan bahwa “The Dripping Tap” tidaklah eksplisit.

Gambar 5.6 Uji coba 4

Sumber: Penulis

VI. KESIMPULAN

Algoritma *string matching* dapat digunakan untuk mendeteksi apakah sebuah lagu mengandung konten eksplisit atau tidak. Dengan menggunakan algoritma *string matching*, kata-kata kasar yang berada di dalam lirik lagu dapat dideteksi asalkan kata-kata kasar tersebut sudah didefinisikan dalam daftar kata kasar terlebih dahulu. Berdasarkan uji coba yang dilakukan oleh penulis, program penulis yang menggunakan algoritma *string matching* sudah dapat mendeteksi lagu yang eksplisit dengan cukup baik.

LINK VIDEO PADA YOUTUBE

https://youtu.be/kRs7h1HylPA

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena atas berkat dan karunia-Nya lah penulis dapat menyelesaikan makalah “Penerapan *String Matching* untuk Mendeteksi Lagu yang Eksplisit” dengan tepat waktu. Penulis juga mengucapkan terima kasih kepada orang tua penulis yang selalu mendukung penulis selama masa perkuliahan penulis di ITB. Terakhir, penulis ingin mengucapkan terima kasih kepada ibu Masayu Leylia Khodra, ibu Nur Ulfa Maulidevi, dan bapak Rinaldi Munir yang telah membimbing penulis selama satu semester ini dalam menjalankan kelas IF2211 Strategi Algoritma.

REFERENSI

[1] Munir, Rinaldi. 2021. Pencocokan String (*String/Pattern Matching*). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf, diakses 20 Mei 2022.

[2] Munir, Rinaldi. 2021. String Matching dengan Regular Expression. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf, diakses 20 Mei 2022.

[3] https://www.dictionary.com/browse/music, diakses 20 Mei 2022.

[4] https://community.spotify.com/t5/Content-Questions/Explicit-content/td-p/4625150#:~:text=The%20explicit%20logo%20is%20applied,unsuitable%20for%20children%3A%20Strong%20language., diakses 20 Mei 2022.

[5] https://genius.com/, diakses 21 Mei 2022.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2022

Jevant Jedidia Augustine - 13520133